PROGRAM PLOTTAB: A Code Designed to Plot Continuous and/or Discrete Physical Data
(VERSION 2000-1)

Abstract

PLOTTAB is designed as a general purpose plotting utility code to plot continuous and/or discrete physical data for use in almost any application. It is designed to be easily used by your application codes to produce your output results in a form that can be immediately used by PLOTTAB to allow you to see your results.

It produces on screen graphics as well as Postscript formatted output files that can be viewed or printed on any Postscript printer. The code is designed to be easily used on any computer - not only today's computers, but also anything that comes along in the future. So you can be assured that once you start using PLOTTAB your graphics problems are over - not just today, but well into the future.

Introduction

PLOTTAB is designed as a simple plotting code that can be used on virtually any computer and graphics device to plot continuous and/or discrete physical data for use in almost any application. It is designed to be easily used by your application codes to produce your output results in a form that can be immediately used by PLOTTAB to allow you to see your results.

It produces on screen graphics as well as Postscript formatted output files that can be viewed or printed on any Postscript printer. The code is designed to be easily used on any computer - not only today's computers, but also anything that comes along in the future.

Only Do the Job Once

Best of all you can be assured that unlike other computer graphics codes that quickly come and go, PLOTTAB will be here not only today to meet your needs, but also into the future to meet your needs. The code has been conservatively designed to not only run on virtually any of today's computers, but also be to easily implemented on any new computers that come along in the future. I have now been using PLOTTAB and its predecessors for almost 20 years, and during that time as each new computer has come along PLOTTAB has been a complete plug-in that has smoothly and effortlessly moved from one computer to the next.

With this approach you can be assured that once you start using PLOTTAB your plotting problems are over, not only for today, but also into the future. You only have to do the job once to modify your codes to produce output in the PLOTTAB input format, and then you can be assured that you will be able to produce graphic output well into the future.

Concentrate on Your Applications

You will find that in order to use PLOTTAB you do not have to be a graphics expert, nor do you have to spend much time learning how to use the code. This allows you to concentrate on your applications, instead of worrying about how you are going to plot your results. Once you have started using PLOTTAB you can take graphics output for granted; something that you never have to worry about or spend much time on ever again. It will simply always be there when you need it.

What Computer Can You USE?

The non-interactive version can be run on ANY computer. The non-interactive code is written in standard FORTRAN and outputs standard formatted Postscript files that can be printed on any Postscript printer, or viewed with any Postscript viewer, such as Ghostview.

The interactive version can be run on UNIX, IBM-PC, Power MAC and even Laptop computers. The interactive code is identical to the non-interactive code, except that a very simple graphics interface for on screen graphics is loaded with the code. The code is distributed with graphic interfaces for UNIX, IBM-PC, and Power MAC. However, it should be very simple to interface this code for on screen graphics on any computer, e.g., it took me 3 days to write the IBM-PC interface and 2 days for the Power MAC version.

Reading and Interpreting Data

All data is read by this code in character form and internally translated into integer or floating point form as needed. This means: 1) it's difficult to get the code to crash by improperly defining input. 2) your input can be in quite general form and will still be properly interpreted by the code, e.g., 14, 1.4+1, 14E+00, 1.4D+01, are all 14 as far as this code is concerned. 3) the code can distinguish between **BLANK** and **ZERO** input - **WARNING** - when this documentation says **BLANK**, it means **BLANK** - in this case nothing else will be interpreted correctly as input.

Data Formats

The formats of the continuous and discrete physical data read by this code are designed to be very simple, so that any of your computer codes can be simply modified to produce output results in the PLOTTAB input format.

Continuous Data Format

The continuous data includes a one line title, followed by a series of (x,y) coordinates, one per line. Each "curve" of continuous data is terminated by a **blank** line. One curve can be followed by another, starting with the one line title, another followed by a series of (x,y) coordinates and terminated by a **blank** line. The input to this code can include any number of such "curves", one curve after the other in the continuous data input file **PLOTTAB.CUR**. Each pair of (x,y) coordinates are in fixed fields each 11 columns wide, corresponding to **FORTRAN 2E11.4 format**. For example,

```
Example Curve # 1
 17      43.0
 19      37.0
      .
      .
 71      12.9
                    (BLANK LINE ENDS CURVE)
Example Curve # 2
      .
      .
```

WARNING - again, I'll stress that a **BLANK LINE** means **completely blank** in the first 22 columns - **NOT zero**, which is considered by the code to be perfectly legal input as part of a "curve".

Discrete Data Format

The format of the discrete data is very similar to the continuous data. Each set of discrete points starts with a one line title, followed by a series of points, and ends with a **blank** line. Each point is defined by an (x,y) coordinate plus uncertainties in both x and y - each point is defined by six values: x, -dx, +dx, y, -dy and +dy, one point per input line. The input to this code may include any number of sets of discrete points, one set after the other in the discrete data input file **PLOTTAB.PNT**. Each point of six values is in fixed fields each 11 columns wide, corresponding to **FORTRAN 6E11.4 format**. For example,

```
Example Set # 1
 17      1.2     2.4   43.0    17.2   12.1
 19      1.6     2.6   37.0    15.8    9.3
      .
      .
 71      8.2    10.7   12.9     7.2    2.3
                    (BLANK LINE ENDS CURVE)
Example Set # 2
      .
      .
```

Note, uncertainties -dx, +dx, -dy and +dy are always interpreted as positive (sign ignored) in the same units as the data (not % or fraction or anything else), and they are interpreted x +/- dx - not a minimum, average and maximum, e.g., for the first x value 17 1.2 2.4, means an average value of 17 with errors extending -1.2 below 17, and +2.4 above 17 - there are no implied units - everything is absolute - below you will learn how to physically interpret and identify your data.

Note, this format allows for zero, or blank, error fields, as well as asymmetric errors. If the errors are symmetric you must define both of them separately.

.

WARNING - again, I'll stress that a **BLANK LINE** means **completely blank** in the first 66 columns - **NOT zero**, which is considered by the code to be perfectly legal input as part of a set of discrete points.

Interpretation of the Data

When you generate data in these formats using one of your application codes you do not have to decide in advance how they will actually be interpreted and appear on a plot. This is controlled by an input file named **PLOTTAB.INP**. This file defines how many curves and/or sets of discrete points will appear on each plot, allows for a two line title to appear at the top of each plot, x and y axis labels and scaling (linear or log scaling), and x and y ranges (in case you do not want to plot all of the data on a single plot). Using these options you can customize each plot to exactly meet your needs. As in the case of the continuous and discrete point files, almost all input fields in **PLOTTAB.INP** are 11 columns wide corresponding to **FORTRAN E11.4 or I11 format**.

How Do You Produce Output

Let's first see how simple it is to update any of your codes to produce output that can be read as input to PLOTTAB. Assume that you have a code that is calculating lots of results that until now you have had to wade through by hand to see what your results mean. Here's what you add to your code to output the results in a form that PLOTTAB can read as input,

```
C-----LOOP OVER CURVES
      DO ICURVE = 1,NCURVE
C-----PRINT FIRST LINE TITLE
      WRITE(16,1600) TITLE(ICURVE)
 1600 FORMAT(A40)
C-----PRINT DATA POINTS
      DO IPOINT = 1,NPOINT(ICURVE)
      WRITE(16,1610) X(IPOINT,ICURVE),Y(IPOINT,ICURVE)
 1610 FORMAT(2E11.4)
C-----END OF POINT LOOP
      ENDDO
C-----PRINT BLANK LINE FOR END OF CURVE
      WRITE(16,1620)
 1620 FORMAT(30X,'(BLANK LINE ENDS CURVE)')
C-----END OF CURVE LOOP
      ENDDO
```

That's all you have to add to your application codes. We just output any number of curves (defined by **NCURVE**), each with its own title line to identify it (defined by **TITLE**), each with any number of points defining each curve (defined by **X, Y**), and each curve ended with a **blank** line. What could be simpler? The above example is for outputting continuous curves, but outputting discrete sets of points is no more difficult. The only difference in the above coding would be where we output two numbers for each (x,y) point, for discrete points we would output six numbers to include the x and y uncertainty in the order x -dx +dx y -dy +dy.

That's it!!! You are now an expert at producing output that can immediately be read by PLOTTAB to show you your results. That's all you need to know about producing output.

Once you start using this very simple approach never again you will have to wade through piles of results trying to figure out what they mean. Instead you can immediately see your results, and you will see them on plots that are of high enough quality to be accepted by journals for publication without any modifications.

Interpreting and Plotting Your Data

Let's assume that you have now produced some output results that you want to plot, i.e., you have your results in the correct format, described above, in a file named **PLOTTAB.CUR** for continuous curves and in **PLOTTAB.PNT** for discrete points. First let me make it clear that you don't need both; you can plot continuous and/or discrete data in any combination. For the following example I'll assume both merely so that I can discuss how to control both.

Now we will edit the control file **PLOTTAB.INP** to tell PLOTTAB how to interpret and plot your data. Below is an example **PLOTTAB.INP** that I recently used to produce a plot. At first this may look complicated, but let me point out that although I have now been using PLOTTAB and its predecessors for almost 20 years, I still have only one **PLOTTAB.INP** and every time I want to produce a plot all I do is edit a few things and bingo! Out come the plots I want. This is the approach that I suggest you also use - don't start from scratch - start with the PLOTTAB.INP file distributed with the code and modify it to meet your needs.

```
    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
```

Let me first cover the input things that I change for each new plot that I want. Above are eight (8) lines from PLOTTAB.INP that are used to produce one plot and I'll discuss the important parameters from top to bottom. I've highlighted the parameters I will discuss so that you can more easily find them.

The first important thing to tell PLOTTAB is how many continuous curves and/or discrete sets of points to put on the plot. This is done on the second line, where my input says to display **3** continuous curves and **2** sets of discrete data. Regardless of how many curves and sets of points you have produced in PLOTTAB.CUR and PLOTTAB.PNT you can use these 2 input fields to tell PLOTTAB how many to actually read and put on the next plot. For example, I can use the above input to produce a plot with 3 curves on the plot, or I can easily change the input (as we will see below) to display the curves one at a time on a series of plots.

Next we want to physically describe what data we are plotting. This is done using the next four (4) input lines. These are,

1) A label for the x axis - in this example `Neutron Energy (MeV)`

2) A label for the y axis - in this example `Cross Section (barns)`
3-4) A two line title to appear at the top of the plot - in this example

```
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
```

In most cases that's all I modify - the code takes care of everything else - so I can immediately run PLOTTAB and get the plots I want. That's all you need to know to successfully use PLOTTAB to generate plots for you.

Note, with this approach the code does not have to have any idea what the physical significance of the data is, and any data can be put into the PLOTTAB input format and plotted. Physical interpretation of the data is all in your hands - by changing plot titles and axis labels you are free to interpret the data any way that you see fit - and you can easily produce plots to specifically meet your needs.

Let me briefly cover the meaning of the other input fields, just in case you want to get fancy.

The first input line, again shown below, defines the (x,y) dimensions of the plot, how many sub-plots to put on each plot, and how large to make the characters. Reading left to right the below line says the plot size is x=0 to 13.5, and y=0 to 10. This will give you a full page plot on 8-1/2 by 11" paper, and except for special purpose you won't want to change this. One case in which you will want to change them is if your plotter doesn't use inches, but rather use cm, mm, anything - no problem - just change these once for your system and you probably will never have to change them again. The next 2 fields say that each page will contain 1 plot in the x direction and 1 y in the y direction = one single, full page plot. If I would like 6 plots on each page I could change these to 3 plots in the x direction and 2 in the y direction. The last field, 1.5, says to make the characters 1-1/2 times normal size.

```
    0.00000   13.50000    0.00000   10.00000          1          1 1.5
```

On the next line you already know about the first two fields. The remaining fields, from left to right, allow you to:
1) add a border around each plot
2) add an (x,y) grid - the code has 6 built-in grids
3) plot the ratio of everything to the first curve
4) change the thickness of all lines drawn

```
          3          2          0          0          0          0
```

You already know about the next four lines, so all we need discuss is the last two lines. These two lines are used to control the x and y features of the plot: the first line is for x and the second for y. Again from left to right, the six (6) fields on each line control,

1) the lower x limit
2) the upper x limit
3) should discrete point x errors be plotted = no(0), yes (1)

4) x scaling = automatic (0), linear (1), log(2)
5) round x limit to avoid touching edge of plot = yes(0), no(1)
6) show legend box = yes(0), no(1)

1) the lower y limit
2) the upper y limit
3) should discrete point y errors be plotted = no(0), yes (1)
4) y scaling = automatic (0), linear (1), log(2)
5) round y limit to avoid touching edge of plot = yes(0), no(1)
6) show data points on curves = no(0), yes(1)

```
                                0          0          0          0
                                0          0          0          0
```

WARNING - let me again stress that this code can tell the difference between **BLANK** and **ZERO** - for example, on the above two input lines the **BLANK** x and y lower and upper limits means scale the plot to show all of the data as read. In contrast, if one of these fields contained **0.0** the specified limit would be forced to be zero, regardless of the range of the data read.

Again, that's it!!!If you want to get fancy you can use these parameters to customize plots to obtain almost any output that you want to meet any specific need.

Multiple Plots

Let's now generalize to more than one plot. Don't worry there isn't much to generalize. For more than one plot you basically have two options:

1) If the layout of each plot is the same you need merely copy the last four lines as many times as you want, changing any parameters that you want on these lines. You can do this as many times as you want for as many plots as you want. For example here's a generalization of our above input for three plots - in this case all I did was copy the last four lines twice and change the titles for Li^{6}, Al^{27} and U^{238} - this assumes that in PLOTTAB.CUR I have 3 curves for each plot (at least 9 curves), and that in PLOTTAB.PNT I have 2 sets of discrete points for each (at least 6 sets).

```
    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
Aluminum-27 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
Uranium-238 Major Cross Sections
```

```
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
```

2) If the layout of each plot is different you can follow the eight (8) line input for one plot by a **BLANK** (not 0, BLANK) line, and then add eight lines for the next plot. You can do this as many times as you want for as many plots as you want. For example here's a generalization of our above input for three plots using this second method - in this case the input is EXACTLY equivalent to what I have shown above using the first method - use either method, or a combination of the two, in any order that you want - the choice is your's.

```
0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0

    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Aluminum-27 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0

    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Uranium-238 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
```

You might wonder why I wrote out the names instead of just using Li^{6}, Al^{27} and U^{238} - if you prefer this form, no problem - see the documentation within the code on how to use sub and super-scripts, as well as Greek characters and other interesting goodies that I cannot cover in detail in a brief introduction.

Data is Read in Order

In controlling the flow of curves from PLOTTAB.CUR and sets of discrete data points from PLOTTAB.PNT remember each time you use input in PLOTTAB.INP to tell it to read data from these files the code continues to read further and further into each file. For example, in the above case the first plot will contain the first three curves from PLOTTAB.CUR and first two sets of points from PLOTTAB.PNT. The second plot will contain curves four through six from PLOTTAB.CUR and sets of points three and four from PLOTTAB.PNT, etc. It's your responsibility to insure that you have properly ordered the curves and sets of points, and properly arrange their grouping on successive plots.

Terminating Plotting

Execution terminates when the are no more requests for plots in PLOTTAB.INP, or no more data to plot from PLOTTAB.CUR and PLOTTAB.PNT. Note, regardless of how many curves and set of points you have in PLOTTAB.CUR and PLOTTAB.PNT only as many plots will be produced as you define in the control file PLOTTAB.INP. Also if one input stream of data (curves or sets of points) is exhausted, but the other isn't, the code will continue producing any plots that you request. For example, for the above three plots if there are 9 curves in PLOTTAB.CUR, but only 2 sets of points in PLOTTAB.PNT, the first plot will contain 3 curves and 2 sets of points (which exhausts the sets of points), and the following two plots will each contain 3 curves and 0 sets of points.

Editing Files

All of the files PLOTTAB.CUR, PLOTTAB.PNT and PLOTTAB.INP are simple text files, so you can use any editor to edit them. For example, after you have produced output in PLOTTAB.CUR if you can want to change anything just open the file and do it. You can change titles for the curves, deletes curves, rearrange the order of curves, anything you want to do to meet your needs.

How Do You Define Input Parameters Before You Have Seen ANY Plots

It very nice to have all of these options to select x and y scaling, x and y ranges, etc. But how can you possibly know what options to select before you have seen plots of you data? The answer is: you don't! PLOTTAB is supplied in two versions: an interactive version that only produces on-screen output, and a non-interactive version that only produces Postscript hardcopy output.

What I recommend is that you start by not selecting any special options and first run the interactive version. This version will allow you to interactively select many of the options described above, e.g., x and y scaling, x range, etc. Then after you have seen your data and played with it you can decide what options you want to set to produce your final Postscript output.

As I use PLOTTAB well over 90 % of the plots that I look at are generated by the interactive code and I never even bother to generate Postscript output. In this way PLOTTAB can be used very simply to quickly look at enormous amounts of data. In this case I don't care what the x and y axis labels are or what the two lines at the top of the plot say - I know how to physically interpret the data I'm looking at - so I just use my existing PLOTTAB.INP file and all I have to tell it is how many curves and sets of discrete points to display on each plot, and I quickly start looking at my results.

Only when I see something of interest that I need a hardcopy of do I bother making any of the changes I have described above. If you also use this pragmatic approach you can save yourself a lot of time and energy and use PLOTTAB much more effectively in your work.

How Do You Remember ALL of the Options

You don't - or at least, I don't. Whenever I want to find out what a given field in the PLOTTAB.INP input parameters means I run PLOTTAB, immediately kill it (use CONTROL C) and then look in the output file PLOTTAB.LST that contains an interpretation of all of the input parameters. For example, for the following input,

```
    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          0          0          1          0          0
Neutron Energy (MeV)
Production, Absorption and Leakage
Production, Absorption and Leakage
For Test Problem
                                0         -2          1          0
                                0         -2          1          0
```

Here's the interpretation of the above input from the output file PLOTTAB.LST. As you can see there is a line by line and field by field interpretation of the input parameters in exactly the order they appear in the input. For example, if I can't remember which input field controls plotting ratios, from the below listing I can see that it is the fifth field of the second input line.

```
========================================================================
 PLOT TABULATED DATA (PLOTTAB VERSION 97-1)
========================================================================
 DESCRIPTION OF PLOTTER AND FRAME LAYOUT
 -----------------------------------------------------------------------
 X DIMENSIONS (X-MIN TO X-MAX).....   .00000+ 0 TO   1.35000+ 1
 Y DIMENSIONS (Y-MIN TO Y-MAX).....   .00000+ 0 TO   1.00000+ 1
 PLOTS PER FRAME (X BY Y)..........          1 BY             1
 CHARACTER SIZE MULTIPLIER.........       1.500
========================================================================
 READ AND PLOT (FOR EACH PLOT).....    3 CURVES
 SETS OF POINTS PER PLOT........... NONE
 SHOULD BORDER BY PLOTTED.......... NO
 TYPE OF GRID......................DASHED GRID (COARSE)
 SHOULD RATIOS BE PLOTTED.......... NO
 LINE THICKNESS....................   0
 -----------------------------------------------------------------------
 X AXIS LABEL AND UNITS............ Neutron Energy (MeV)
 Y AXIS LABEL AND UNITS............ Production, Absorption and Leakage
 -----------------------------------------------------------------------
 -----------------------------------------------------------------------
 PLOT TITLE
........................................................................
 Production, Absorption and Leakage
 For Test Problem
 -----------------------------------------------------------------------
 REQUESTED X RANGE................. PLOT ALL POINTS
 PLOT X ERROR BARS................. NO
 X PLANE ON PLOTS (IF POSSIBLE)....LOG (NO INTERPOLATION)
 ROUND X LIMITS.................... NO
 LEGEND BOX ON PLOT................ YES
 -----------------------------------------------------------------------
 REQUESTED Y RANGE................. PLOT ALL POINTS
 PLOT Y ERROR BARS................. NO
 Y PLANE ON PLOTS (IF POSSIBLE)....LOG (NO INTERPOLATION)
 ROUND Y LIMITS.................... NO
 SHOW DATA POINTS.................. NO
 -----------------------------------------------------------------------
 X LIMITS (PLANE)..................  1.02360-10 TO   1.99760+ 1 (LOG)
 Y LIMITS (PLANE)..................  2.02610- 5 TO   1.57230+ 7 (LOG)
 -----------------------------------------------------------------------
 CONTINUOUS CURVES
........................................................................
 INDEX POINTS DESCRIPTION
........................................................................
     1    566 Production
     2    566 Absorption
     3    498 Leakage
```

Postscript Output Files

When you run the non-interactive version of the code it will produce a series of Postscript output files, one plot per file. The files will be named,
PLOT0001.ps
PLOT0002.ps
.
.
These postscript files can be printed on any Postscript printer, or viewed with any Postscript viewer, such as Ghostview; note, Ghostview is available FREE on the web.

WARNING - every time you run the code it uses the same file names. So if you want to save any file make sure you rename it before you again run the code.

Interpolation Along Curves

When you start using PLOTTAB you will find that your results can look quite different depending upon how you display the data, e.g., linear or log scaling in the x or y direction.

By default PLOTTAB assumes that between tabulated points of curves it should interpolate assuming linear interpolation in x and y. If your data is not linearly interpolable you can get some very strange looking results when PLOTTAB interpolates in say log/log x/y scaled plots. PLOTTAB ALWAYS uses the defined interpolation to show the TRUE shape of each curve between tabulated points in each possible combination of linear and log x and y scaling. For example, if I have only two tabulated points at x=1 and x=100, using standard linear interpolation this will be a straight line on a linearly scaled x and y plot. But if you use log x and/or y scaling PLOTTAB will display a curved line corresponding to the TRUE (assuming linear interpolation) shape of the curve.

You can control how PLOTTAB interpolates in x and y separately by defining the interpolation to be linear or log in each dimension. Above I briefly described how to control the x and y scaling of each plot = automatic (0), linear (1), log(2). In this case, any non-negative values specify scaling with **linear** interpolation. To control interpolation, any negative values specify scaling and the same type of interpolation; the only meaningful negative value is -2 = **log** scaling and interpolation.

An important point to note, is that generally if you are using tabulated data in your applications it is important how the data is interpreted not only at the points that you tabulate data, but AT EVERY SINGLE POINT along the curve, e.g., integrals and interpolate values depend crucially on EXACTLY HOW YOU INTERPOLATE.

This PLOTTAB option can supply you with value information that you can use in your applications. For example, if you assume your data is linearly interpolable, but you see funny bumps and cusps in the results when you plot it in various combinations of linear and log, x and y scaling, you better think again about your assumption, because your data is not smoothly linearly interpolable. In this case you should consider either adding more, closer spaced points along your curves, or try PLOTTAB's log interpolation and see if this solves your problem. It ONLY SOLVES YOUR PROBLEM if you are willing to interpret your data in your applications using more complicated log interpolation. What should you do? The choice is yours. But whatever you do, be sure that you use PLOTTAB to check your final results to insure that you are not incorrectly interpreting your data in your applications.

You are NOW a PLOTTAB Expert

Sorry, that's about all I can quickly teach you about PLOTTAB. If you now understand how to produce output from your codes to be used as input to PLOTTAB, and how to edit PLOTTAB.INP to define the layout of each plot, you are now an expert, and you can immediately start generating plots. If you want to get even fancier, see the following documentation for more details - there's a lot more that this code can do to meet your specific needs than I have been able to cover in this brief introduction.

Code Installation

The code is distributed with detailed instructions concerning installation and testing of the code. These instructions are periodically updated for distribution with the code, to insure that the instructions are as up-to-date as possible, and exactly correspond to the version of the code that you will be implementing and using.

When you receive this code system you will find it arranged in a file directory structure. At each level of the directory you will find a file named **README** - be sure that you read all such files as you proceed with installation and testing.

Additional Documentation

This report is designed to be used on-line and as such has been restricted to simple text in Microsoft Word 5.1 format. In particular a minimum of graphic output is included; only one simple example is included below. For additional documentation, including many example plots see the earlier documentation,

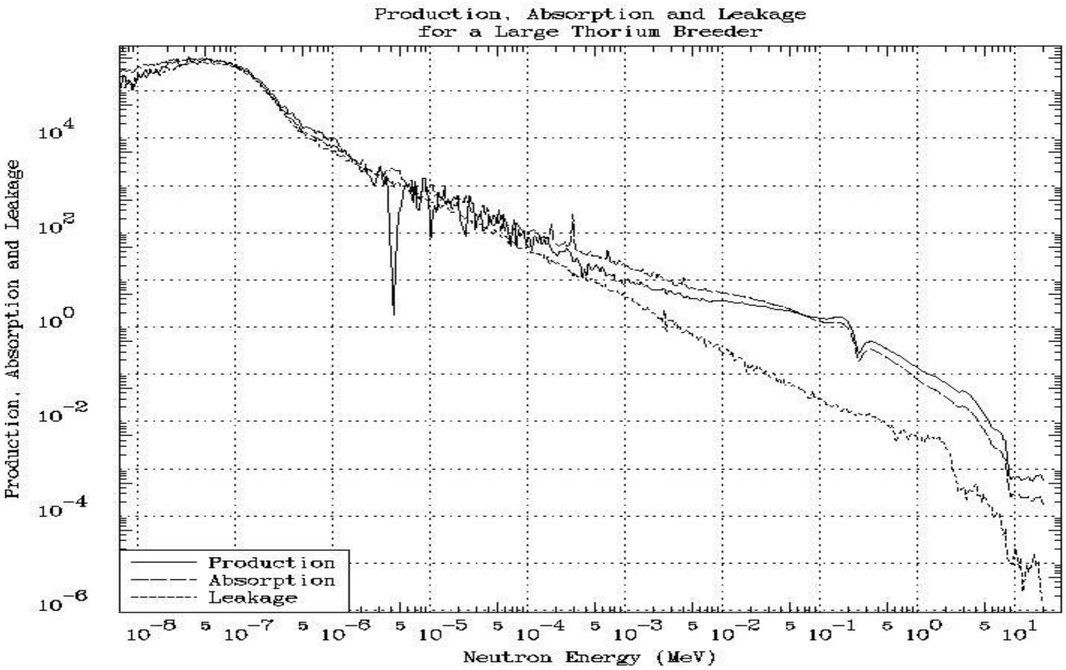
PROGRAM PLOTTAB: A Code Designed to Plot Continuous and/or Discrete Physical Data, by Dermott E. Cullen, UCRL-ID-110240, (March 1992), Lawrence Livermore National Laboratory

Latest Documentation

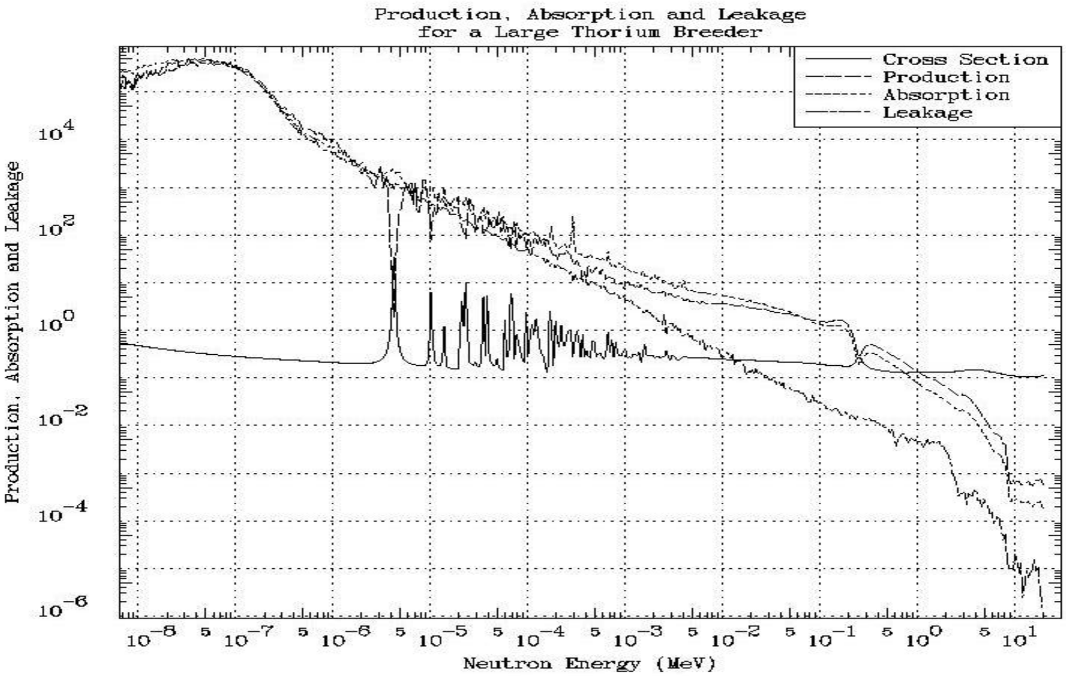
This code is designed to be self-documenting, in the sense that the latest documentation for the code is included as comment lines at the beginning of the code. Printed documentation, such as this report, is periodically published and consists mostly of a copy of the comment lines from the beginning of the code. The user should be aware that the comment lines in the code are continuously updated to reflect the most recent status of the code and these comment lines should always be considered to be the most recent documentation for the code and may supersede published documentation, such as this report. Therefore users are advised to always read the documentation within the actual code.

Example PLOTTAB Problem

When you run the PLOTTAB test problem you will see the two figures discussed here. After running TART2000 the utility code BALANCE was used to read the TART2000 results and output them in the PLOTTAB format. The first figure you see when you run PLOTTAB illustrates the results for the neutron balance of the system: neutron production, absorption and leakage. The results, from higher to lower energy, show a fission spectrum, slowing down spectrum and finally a thermal Maxwellian. Based solely on this figure it is difficult to understand some features of the spectrum, such as the minimum in production at a few eV. Why is there is minimum and why isn't there also a minimum in the absorption at this point? This is explained below.



By adding the total cross section to the second figure, when you run PLOTTAB you can see that the minimum in the production corresponds to a maximum in the cross section due to a capture resonance, i.e., classical self-shielding where an increase in cross section results in a corresponding decrease in flux so that the product is about constant. How do I know this is a capture resonance? Note, the production, but not the absorption has a minimum. Self-shielding theory tells me that since absorption is smooth the product of the flux and absorption cross section must be almost the entire reaction rate. In contrast the minimum in production tells me the fission cross section is smooth, so that the product of the reduced self-shielded flux and smooth cross section results in a minimum in production.



Hopefully these results illustrate how the TART2000 system codes can be used in combination to produce not just numbers, but rather physical understand. By using TARTCHEK to check input, TART2000 to run your calculations, the utility codes to process the results, and PLOTTAB and TARTCHEK to immediately show you your results, you can save a great deal of time and learn a lot more than you ever will by wading through piles of output. For example, in this simple case the TART2000 output file was over 32,000 lines long. But I learned everything I wanted to know about this system without ever having to look at this output. In addition you have EPICSHOW to independently check results compared to the cross sections TART2000 is using.